Amendments to the Claims

Please cancel claims 1, 18, and 25.

LISTING OF CLAIMS

This listing of claims will replace all prior versions, and listing, of claims in the application:

1. ***(Withdrawn)*** A method for coding test pattern for integrated circuits (ICs) in scan design and with build-in test hardware (BIT-HW), whereby the ICs have IC design data, the BIT-HW consists of a linear feedback shift register (LFSR) for pseudo-random pattern generation and a computer program for pattern merging and distribution over scan chains, the method comprising the steps of:

specifying a logic model representation of the physical BIT-HW;

specifying test vectors one by one;

compressing said test vectors one by one; and

providing the compressed test vectors.

2. ***(Currently Amended)*** A method for coding test pattern for integrated circuits (ICs) in scan design and with build-in test hardware (BIT-HW), whereby the ICs have IC design data, the BIT-HW consists of a linear feedback shift register (LFSR) for pseudo-random pattern generation and a computer program for pattern merging and distribution over scan chains, the method comprising the steps of:

~~specifying~~ generating a logic model representation of the physical BIT-HW;

specifying test vectors one by one;

compressing said test vectors one by one; and

providing the compressed test vectors, whereby said step of ~~specifying~~ generating the logic model representation includes a step of ~~specifying~~ building a function operator and an initial state-function from respective IC design data, whereby said

function operator is an executable logic model representation of the compressed test vectors.

3. ***(Currently Amended)*** The method according to claim 2, whereby the step of ~~specifying~~ generating the logic model representation includes a step of specifying a set of chain access operators from respective IC design data, and whereby each chain access operator corresponds to a certain scan chain and is an executable logic model representation of said ~~means~~ computer program for pattern merging and distribution.

4. ***(Currently Amended)*** The method according to claim 2, whereby the step of ~~specifying~~ generating the logic model representation includes a step of generating an LFSR generator code from said initial state-function by an iterative execution of said function operator.

5. ***(Currently Amended)*** The method according to claim 2, whereby the step of ~~specifying~~ generating the logic model representation includes a step of ~~generating~~ building state-functions by building an LFSR generator matrix line by line beginning with said initial state-function by an iterative execution of said function operator.

6. ***(Currently Amended)*** The method according to claim 4, whereby the step of ~~specifying~~ generating the logic model representation includes a step of ~~generating~~ building state-functions by building a set of chain LFSR generator matrices from said LFSR generator code each by iterative executions of the respective chain access operators.

7. ***(Currently Amended)*** The method according to claim 3, whereby the step of ~~specifying~~ generating the logic model representation

includes a step of ~~generating~~ building state-functions by building a set of chain LFSR generator matrices from said initial state-function each by iterative executions of said function operator in combination with the respective chain access operators.

8. ***(Original)*** The method according to claim 4, whereby said LFSR generator code is a binary vector of a length that corresponds to a certain maximum number of execution cycles of said LFSR.

9. ***(Original)*** The method according to claim 5, whereby the LFSR generator matrix is a binary matrix with a line width of the length of the LFSR and a column length of the length of the respective scan chain.

10. ***(Currently Amended)*** The method according to claim 5, whereby said state-function is a binary vector having the length of the LFSR and represents an executable function to be executed by the following action sequence:

i. selecting bit positions of the LFSR in correspondence to the 1-values of the state-function;

ii. combining the 0/1-values of the selected bit positions by XOR arithmetic; and

iii. presenting the XOR-output 0/1-value to ~~the~~ a user or to another system for further processing.

11. ***(Previously Presented)*** The method according to claim 2, whereby the step of specifying a test vector includes a step of specifying a set of care-bits from respective IC design data, specifying for each care-bit a 0/1-value and specifying a position referenced to a specific scan chain and execution cycle in which this 0/1-value is generated by the LFSR.

12. ***(Original)*** The method according claim 5, whereby the step of compressing a test vector includes a step of selecting and providing a collection of state-functions out of the LFSR generator matrix, whereby the step of specifying a test vector includes a step of specifying a set of care-bits from respective IC design data, and whereby said specification of the respective care-bit position is the index in this selection.

13. ***(Original)*** The method according to claim 4, whereby the step of compressing a test vector includes a step of computing and providing a collection of state-functions, whereby the step of specifying a test vector includes a step of specifying a set of care-bits from respective IC design data, and whereby said specification of a care-bit position is the index for selecting a sequence-vector out of the LFSR generator code and for selecting a chain access operator for this computing.

14. ***(Original)*** The method according to claim 12, whereby said step of compressing a test vector includes a step of solving a linear equation system formed by said collection of state-functions, whereby the solution is a compressed LFSR-Code.

15. ***(Original)*** The method according to claim 14, whereby the step of compressing a test vector and solving a linear equation system comprises a method for computing the LFSR-Code in a variable length code, whereby the method provides a special solution characterized by the longest left or right adjusted uniform sequence of either zeros or ones.

16. ***(Original)*** The method according to claim 15, whereby the step of providing the compressed test vectors in a variable length code includes a step of sorting and grouping the total of compressed test vectors over the individual lengths and storing

them in data records having suitable formats for each length interval.

17. ***(Previously Presented)*** The method according to claim 2, whereby the step of providing the compressed test vectors includes a step of simulating a test execution using the compressed test vectors and providing the simulation results to the user or another system for further processing.

18. ***(Withdrawn)*** A system for coding test pattern for integrated circuits (ICs) in scan design, said system comprising:

build-in test hardware (BIT-HW), whereby the ICs have IC design data;

a linear feedback shift register (LFSR) included in said BIT-HW for pseudo-random pattern generation; and

a computer system including instructions to execute a method for pattern merging and distribution over scan chains, said method comprising the steps of:

specifying a logic model representation of the physical BIT-HW;

specifying test vectors one by one;

compressing said test vectors one by one; and

providing the compressed test vectors.

19. ***(Currently Amended)*** A system for coding test pattern for integrated circuits (ICs) in scan design, said system comprising:

build-in test hardware (BIT-HW), whereby the ICs have IC design data;

a linear feedback shift register (LFSR) included in said BIT-HW for pseudo-random pattern generation; and

a computer system including instructions to execute a method for pattern merging and distribution over scan chains, said method comprising the steps of:

~~Specifying~~ generating a logic model representation of the physical BIT-HW;

specifying test vectors one by one;

compressing said test vectors one by one; and

providing the compressed test vectors, whereby said method step of ~~specifying~~ generating the logic model representation includes a step of ~~specifying~~ building a function operator and an initial state-function from respective IC design data, whereby said function operator is an executable logic model representation of the compressed test vectors.

20. ***(Currently Amended)*** The system according to claim 19, whereby the method step of ~~specifying~~ generating the logic model representation includes a step of specifying a set of chain access operators from respective IC design data, and whereby each chain access operator corresponds to a certain scan chain and is an executable logic model representation of said ~~means~~ method for pattern merging and distribution.

21. ***(Currently Amended)*** The system according to claim 19, whereby the method step of ~~specifying~~ generating the logic model representation includes a step of generating an LFSR generator code from said initial state-function by an iterative execution of said function operator.

22. ***(Currently Amended)*** The system according to claim 19, whereby the method step of ~~specifying~~ generating the logic model representation includes a step of ~~generating~~ building state-functions by building an LFSR generator matrix line by line beginning with said initial state-function by an iterative execution of said function operator.

23. ***(Currently Amended)*** The system according to claim 20, whereby the method step of ~~specifying~~ generating the logic model representation includes a step of ~~generating~~ building state-functions by building a set of chain LFSR generator matrices from said LFSR generator code each by iterative executions of the respective chain access operators.

24. ***(Currently Amended)*** The system according to claim 20, whereby the method step of ~~specifying~~ generating the logic model representation includes a step of ~~generating~~ building state-functions by building a set of chain LFSR generator matrices from said initial state-function each by iterative executions of said function operator in combination with the respective chain access operators.

25. ***(Withdrawn)*** A program product for coding test pattern for integrated circuits (ICs) in scan design and with build-in test hardware (BIT-HW), whereby the ICs have IC design data, the BIT-HW consists of a linear feedback shift register (LFSR) for pseudo-random pattern generation, the program product comprising:

a computer readable medium having recorded thereon computer readable program code for performing a method of pattern merging and distribution over scan chains, said method comprising the steps of:

specifying a logic model representation of the physical BIT-HW;

specifying test vectors one by one;

compressing said test vectors one by one; and

providing the compressed test vectors.

26. ***(Currently Amended)*** A program product for coding test pattern for integrated circuits (ICs) in scan design and with build-in test hardware (BIT-HW), whereby the ICs have IC design

data, the BIT-HW consists of a linear feedback shift register (LFSR) for pseudo-random pattern generation, the program product comprising:

a computer readable medium having recorded thereon computer readable program code for performing a method of pattern merging and distribution over scan chains, said method comprising the steps of:

~~Specifying~~ generating a logic model representation of the physical BIT-HW;

specifying test vectors one by one;

compressing said test vectors one by one; and

providing the compressed test vectors, whereby said method step of ~~specifying~~ generating the logic model representation includes a step of ~~specifying~~ building a function operator and an initial state-function from respective IC design data, whereby said function operator is an executable logic model representation of the compressed test vectors.

27. ***(Currently Amended)*** The program product according to claim 26, whereby the step of ~~specifying~~ generating the logic model representation includes a step of specifying a set of chain access operators from respective IC design data, and whereby each chain access operator corresponds to a certain scan chain and is an executable logic model representation of said ~~means~~ method for pattern merging and distribution.

28. ***(Currently Amended)*** The program product according to claim 26, whereby the method step of ~~specifying~~ generating the logic model representation includes a step of generating an LFSR generator code from said initial state-function by an iterative execution of said function operator.

29. ***(Currently Amended)*** The program product according to claim 26, whereby the method step of ~~specifying~~ generating the logic model representation includes a step of ~~generating~~ building state-functions by building an LFSR generator matrix line by line beginning with said initial state-function by an iterative execution of said function operator.

30. ***(Currently Amended)*** The program product according to claim 28, whereby the method step of ~~specifying~~ generating the logic model representation includes a step of ~~generating~~ building state-functions by building a set of chain LFSR generator matrices from said LFSR generator code each by iterative executions of the respective chain access operators.

31. ***(Currently Amended)*** The program product according to claim 27, whereby the method step of ~~specifying~~ generating the logic model representation includes a step of building ~~generating~~ state-functions by building a set of chain LFSR generator matrices from said initial state-function each by iterative executions of said function operator in combination with the respective chain access operators.